(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平10－78973

(43)公開日　平成10年(1998)３月24日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ　17/50			Ｇ０６Ｆ　15/60	６６４Ｌ	
17/00			15/20	Ｄ	
			15/60	６６４Ｍ	

審査請求　未請求　請求項の数６　ＯＬ　（全 17 頁）

(21)出願番号　特願平8－232305

(22)出願日　平成８年(1996)９月２日

(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号

(72)発明者　長井　寛志
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内

(74)代理人　弁理士　真田　有

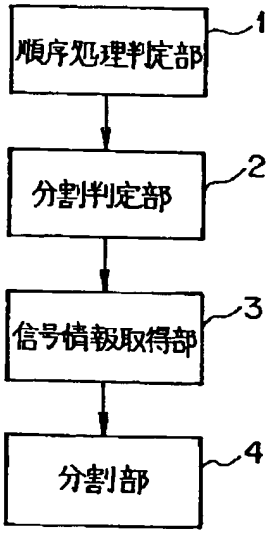
(54)【発明の名称】　順序処理記述の変換方法および装置

(57)【要約】

【課題】　順序処理記述をできる限り同時処理記述に変換し、ＨＤＬで記述された回路の動作を並列処理によりシミュレートする際の処理の高速化をはかる。

【解決手段】　ＨＤＬで記述された順序処理記述を同時処理記述に変換する際に、順序処理記述の中から、信号変化に応じて同期処理される部分を同期部として抽出するとともに、その同期部内において信号代入を行なう部分を信号代入部として抽出し、その信号代入部を順序処理記述から分割するように構成する。

本発明の原理ブロック図



1

【特許請求の範囲】

【請求項1】　ハードウェア記述言語で記述された順序処理記述を同時処理記述に変換するための変換方法において、

前記順序処理記述の中から、信号変化に応じて同期処理される部分を同期部として抽出するとともに、

抽出された前記同期部内において信号代入を行なう部分を信号代入部として抽出し、

抽出された前記信号代入部を前記順序処理記述から分割することを特徴とする、順序処理記述の変換方法。

【請求項2】　前記順序処理記述から分割された前記信号代入部を、その信号代入部と同等の動作を行なうハードウェア素子に置き換えることを特徴とする、請求項1記載の順序処理記述の変換方法。

【請求項3】　前記順序処理記述から分割された前記信号代入部を、その信号代入部と同等の動作を行なうソフトウェアに置き換えることを特徴とする、請求項1記載の順序処理記述の変換方法。

【請求項4】　ハードウェア記述言語で記述された順序処理記述を同時処理記述に変換するための変換装置であって、

前記ハードウェア記述言語による記述中における順序処理記述の有無を判定する順序処理判定部と、

該順序処理判定部により順序処理記述が有ると判定された場合に、その順序処理記述の中に信号変化に応じて同期処理される部分が存在することを条件にして前記順序処理記述を分割可能か否かを判定し、その部分を同期部として抽出する分割判定部と、

該分割判定部により抽出された前記同期部内において信号代入を行なう部分を信号代入部として抽出し、その信号代入部についての信号情報を取得する信号情報取得部と、

該信号情報取得部により抽出された前記信号代入部を、前記順序処理記述から分割する分割部とをそなえたことを特徴とする、順序処理記述の変換装置。

【請求項5】　該信号情報取得部により取得された前記信号代入部についての信号情報に基づいて、該分割部により分割された前記信号代入部を、所定の動作を行なうハードウェア素子に置き換え可能か否かを判定する変換判定部と、

該変換判定部により置き換え可能であると判定された場合に、該分割部により分割された前記信号代入部を、前記ハードウェア素子に置き換える変換実行部とをそなえたことを特徴とする、請求項4記載の順序処理記述の変換装置。

【請求項6】　該信号情報取得部により取得された前記信号代入部についての信号情報に基づいて、該分割部により分割された前記信号代入部を、所定の動作を行なうソフトウェアに置き換え可能か否かを判定する変換判定部と、

2

該変換判定部により置き換え可能であると判定された場合に、該分割部により分割された前記信号代入部を、前記ソフトウェアに置き換える変換実行部とをそなえたことを特徴とする、請求項4記載の順序処理記述の変換装置。

【発明の詳細な説明】

【0001】（目次）

発明の属する技術分野

従来の技術（図16）

発明が解決しようとする課題（図16，図17）

課題を解決するための手段（図1）

発明の実施の形態（図2～図15）

発明の効果

【0002】

【発明の属する技術分野】本発明は、ハードウェア記述言語〔以下、HDL（Hardware Description Language）という〕で記述された回路の動作を並列処理によってシミュレートする際に用いて好適の方法および装置に関し、特に、HDLで記述された順序処理記述を同時処理記述に変換するための変換方法および装置に関する。

【0003】HDLを用いた回路の設計は、設計変更の容易さや理解し易さなどの面から回路を設計する場合には非常に有効であり、回路規模が大きくなるにつれてさらにその重要性が増している。HDLのシミュレーションを行なうことにより実際に回路を作成する前にその動作を確認することができるため、シミュレーションの高速化は設計期間を短縮する上で重要な要素といえる。シミュレーションの高速化としてはHDLの言語の性質上、並列処理が有効である。

【0004】しかし、パイプライン処理や並列処理でシミュレートする場合、順序処理文が並列処理の妨げになりシミュレーションの速度を低下させている。これは、1周期のシミュレーションで同時処理文は1つの処理をシミュレートするのに対し、順序処理文は複数の処理を行なうためである。順序処理文と同時処理文とを並列処理する場合、ある処理部において同時処理は終了していても、その処理部は、他の処理部において順序処理が終了するまで同期待ちを行なう必要があり、効率のよい並列処理を行なえない。つまり、シミュレーション時間の長い順序処理文が、並列度を低下させ、シミュレーション高速化の妨げになっている。

【0005】その解決策としては、順序処理文を同時処理文に変換すること（以下、同時処理化と呼ぶ場合もある）が考えられ、多くの順序処理文は同時実行可能（同時処理化可能）である。しかし、順序回路であるフリップフロップを想定した順序処理文では、信号の変化（立ち上がり／立ち下がり；クロックエッジ）を条件に処理を行なっている。この場合、順序処理の多くを占めるフリップフロップはクロックエッジの判定と信号代入との依存関係を保てなくなるため、順序回路（フリップフロ

ップ）をそのまま直接的に同時処理化することができず、何らかの操作が必要になる。

【0006】

【従来の技術】HDLにより記述された文には、他の文と独立して同時に処理することが可能な同時処理文と、記述された順序で処理されなければならない順序処理文とがある。VHDL〔VHSIC (Very High Speed IC) Hardware Description Language)〕による順序処理文および同時処理文の具体例を図16に示す。図16における文（1）が順序処理文であり、文（2）および（3）が同時処理文である。これらの文（1）～（3）はそれぞれ同時に動作するが、文（1）は一連の文の動作が文（2）および（3）の各文と同じ処理単位として扱われる。なお、順序処理と順次処理とは同義である。

【0007】十分な数の並列処理が可能なシミュレーションで同時処理文または順序処理文を1文だけシミュレートするのに要する時間を1として考え、図16に示す文（1）～（3）をシミュレートした場合、1文で構成される文（2）および（3）のシミュレーションはそれぞれ時間1で終了するが、文（1）は3つの文から成るため、そのシミュレーション時間は3になり、文（1）～（3）を並列処理した場合にHDL全体としてのシミュレーション時間は3になる。

【0008】つまり、順序処理文の処理によってシミュレーション速度が決定してしまうことになる。そのため、並列処理によるシミュレーション高速化を行なうには、順序処理のシミュレーション時間を短くする必要がある。順序処理のシミュレーション時間を短くする一手法としては、前述した、順序処理文の同時処理化が考えられる。この手法は、順序処理で記述された回路が、例えば図16に示す文（1）を成す3つの文のように、全て組合せ回路であるならば、各々の接続関係を保てば同＊

＊時処理を行なってもシミュレーション結果は変わらないという原理を用いたものである。順序処理に、このような変換（同時処理化）を施せば、普通は順序通りに処理される各文を並列に処理できるようになるため、シミュレーション時間が1で済むことになる。

【0009】

【発明が解決しようとする課題】組合せ回路で表せるような順序処理記述は同時処理するように変換できる。しかし、順序処理記述中に信号の変化（クロックエッジ）を判断して処理を行なう記述があると、組合せ回路で表せないため、その順序処理記述を同時処理化することができない。

【0010】例えば、前述した図16に示す順序処理文（1）や、図17（a）に示す順序処理文は、クロックclk の変化に関係しないので、その順序処理文を同時処理文に変換することができるのに対し、図17（b）に示す順序処理文は、クロックclk の立ち上がりで実行されるため、その順序処理文を同時処理文に変換することはできない。なお、図17（b）に示す順序処理文は、クロックclk の立ち上がりで実行されるもので、その記述中、“clk'event ”は、クロックclk が変化することを示している。

【0011】順序処理記述では、同時処理に変換できる組合せ回路に比べ、フリップフロップの数がかなり多い場合がある。予め準備された2種類の回路（No.1, No.2）についての順序処理記述の内容を調べた結果を表1に示す。ただし、表1中において、（　）内は同時処理化不可能な記述のうちのフリップフロップの割合である。

【0012】

【表1】

順序処理記述の内訳（%）

回路の種類	No.1	No.2
同時処理化可能な記述	14	25
同時処理化不可能な記述	86	75
（フリップフロップ）	（60）	（64）

【0013】このようにフリップフロップを多く含む順序処理記述において、組合せ回路についての記述を同時処理記述に変換しただけでは、同時処理化によるシミュレーションの高速化効果をそれほど期待できない。このような回路のシミュレーションを高速化するには、フリップフロップをも同時処理化する必要がある。しかし、フリップフロップは、クロックの立ち上がりや立ち下がり（クロックエッジ）に応じて処理を行なう順序回路であり、このようなフリップフロップを単純に同時処理化することができない。従って、信号変化に依存するため※

※に同時処理記述に変換できない順序処理記述は、変換されることなく、そのまま順序処理されざるを得ず、HDLシミュレーションの並列度を低下させる要因になっていた。

【0014】本発明は、このような課題に鑑み創案されたもので、順序処理記述をできる限り同時処理記述に変換して、HDLで記述された回路の動作を並列処理によりシミュレートする際の処理の高速化をはかった、順序処理記述の変換方法および装置を提供することを目的とする。

【0015】

【課題を解決するための手段】このため、本発明の順序処理記述の変換方法は、HDLで記述された順序処理記述を同時処理記述に変換するべく、順序処理記述の中から、信号変化に応じて同期処理される部分を同期部として抽出するとともに、その同期部内において信号代入を行なう部分を信号代入部として抽出し、さらに、その信号代入部を順序処理記述から分割することを特徴としている（請求項1）。

【0016】そして、順序処理記述から分割された信号代入部を、その信号代入部と同等の動作を行なうハードウェア素子またはソフトウェアに置き換えてもよい（請求項2，3）。一方、図1は、本発明の順序処理記述の変換装置を示す原理ブロック図で、この図1に示すように、本発明の順序処理記述の変換装置は、HDLで記述された順序処理記述を同時処理記述に変換するためのものであって、順序処理判定部1，分割判定部2，信号情報取得部3および分割部4を有して構成されている。

【0017】ここで、順序処理判定部1は、HDLによる記述中における順序処理記述の有無を判定するものであり、分割判定部2は、順序処理判定部1により順序処理記述が有ると判定された場合に、その順序処理記述の中に信号変化に応じて同期処理される部分が存在することを条件にして順序処理記述を分割可能か否かを判定し、その部分を同期部として抽出するものである。

【0018】また、信号情報取得部3は、分割判定部2により抽出された同期部内において信号代入を行なう部分を信号代入部として抽出し、その信号代入部についての信号情報を取得するものであり、分割部4は、信号情報取得部3により抽出された信号代入部を、順序処理記述から分割するものである（請求項4）。さらに、信号情報取得部3により取得された信号代入部についての信号情報に基づいて、分割部4により分割された信号代入部を、所定の動作を行なうハードウェア素子またはソフトウェアに置き換え可能か否かを判定する変換判定部と、この変換判定部により置き換え可能であると判定された場合に分割部4により分割された信号代入部をハードウェア素子またはソフトウェアに置き換える変換実行部とをそなえる（請求項5，6）。

【0019】ところで、順序処理記述の中に信号変化を判定して同期処理を行なうような記述が含まれている場合、その記述を同時処理化することができない。しかし、実際に同時処理化することができないのは信号代入部であり、それ以外の部分については同期処理化は可能である。そこで、本発明では、信号変化に応じて同期処理される同期部内から信号代入部を抽出して分割することにより、順序処理記述を、信号代入部以外の部分については同時処理化可能な形に変換している。このように順序処理記述を信号代入部とそれ以外の部分とに分割して取り扱うことで、順序処理記述の長さを短くできる。

また、信号変化に伴う処理は信号代入部だけになり、それ以外の部分は、分割後、組合せ回路になるので、同時処理化することができる（請求項1，4）。

【0020】さらに、順序処理記述として残る信号代入部を、その信号代入部と同等の動作を行なうハードウェア素子もしくはソフトウェアに置き換えることにより、シミュレーションに際して順序処理を行なう必要がなくなる（請求項2，3，5，6）。

【0021】

【発明の実施の形態】以下、図面を参照して本発明の実施の形態を説明する。図2は本発明の一実施形態としての順序処理記述の変換方法の全体的処理フローおよびその変換装置の構成を示す図である。この図2では、本実施形態による変換方法を成す概略的な6つのステップ11～16が処理順序に従って示されている。そして、これらのステップ11～16が、それぞれ、本実施形態の変換装置を機能的に構成する要素（順序処理判定部，分割判定部，信号情報取得部，分割部，変換判定部および変換実行部）に対応している。

【0022】変換装置は、HDLで記述された順序処理記述を同時処理記述に変換するためのもので、実際にはハードウェアとしてCPUやメモリを有する一般的なプロセッサにより構成されている。そして、CPUが、後述する回路データベースに基づいて、図3～図8に示すフローチャートに従うプログラムを実行することにより、各要素（ステップ11～16）がソフトウェアにより実現されるようになっている。

【0023】本実施形態では、処理対象となるデータとして、HDLで記述されたものを直接的に扱うのではなく、HDLで記述された回路の構造をデータベース化した回路データベース（HDLデータベース）を用いる。以下では、本実施形態による処理手順を説明するとともに、その処理手順を図9に示すような具体的な回路データベースに適用した場合についても合わせて説明する。

【0024】ここで、回路データベース（HDLデータベース）の最小単位である各ブロックには、図9～図11に示すように、そのブロックの識別子（id）とそのブロックの名前（name）とが付されている。各ブロックの識別子（id）はハイフンを挟んだ前後一対の数字として表記されており、最初の数字はそのブロックの属性（1：信号，2：順序処理，3：同時処理）を示すとともに、後ろの数字はそのブロックの属性についての序数を示している。

【0025】各ブロックの名前は、そのブロックの属性が信号ならば信号名〔図9～図11では、CLK(クロック),D,E,P, P´〕とし、そのブロックの属性が順序処理または同時処理ならば機能名とする。なお、図9～図11では、具体的な機能名として"RISE","NOT","OR","ASSIGN","END","FLIPFLOP" が表記されている。"RISE"は信号の立ち上がりを調べるブロックの機能名であり、"NO

T" は信号の否定を行なうブロックの機能名であり、"OR"は信号の論理和を出力するブロックの機能名であり、"ASSIGN"は信号代入を行なうブロック（信号代入部）の機能名であり、"END" は終了を示すブロックの機能名であり、"FLIPFLOP"はフリップフロップとしての機能を果たすブロックの機能名である。

【００２６】また、図9～図１１において、ブロック相互間を結ぶ実線の矢印はデータの流れ（入出力方向）を示し、ブロック相互間を結ぶ点線の矢印は順序処理文（逐次実行文）の実行順序を示している。本実施形態で使用される各種データ構造は、以下の通りである。例えば、“SYNC_PART”の“SIGNAL”のデータは、“SYNC_PART.SIGNAL"として表される。

【００２７】①SYNC_PART：同期部に関するデータ（例えば図４に示す同期部抽出データテーブル２１に格納されるデータ）

-SIGNAL ：同期部における同期基準となるクロック信号のid

-BODY ：同期部の開始ブロックのid

【００２８】②ASSIGN：信号代入ブロック（信号代入部）に関するデータ（例えば図５，図６に示す信号情報取得データテーブル２２に格納されるデータ）

-SIGNAL ：信号代入ブロックで同期基準となるクロック信号のid

- IN_SIGNAL：信号代入ブロックへの入力信号を送るブロックのid

-OUT_SIGNAL：信号代入ブロックからの出力信号を受けるブロックのid

-ASSIGN ：信号代入ブロックのid

-TEMPORARY：信号代入ブロックを分割すべく挿入する信号（後述）のid

【００２９】③FLIPFLOP：フリップフロップ変換用データ〔例えば図7に示すフリップフロップ変換用データテーブル２３に格納されるデータ；ここでは図１２（ａ）に示すフリップフロップ３１を変換対象とする〕

-SIGNAL ：フリップフロップに対するクロック信号のid

- IN_SIGNAL：フリップフロップへの入力信号を送るブロックのid

-OUT_SIGNAL：フリップフロップからの出力信号を受けるブロックのid

-RESET：フリップフロップに対するリセット信号のid

-SET：フリップフロップに対するセット信号のid

【００３０】さて、ＨＤＬで記述された順序処理記述を同時処理記述に変換するために、本実施形態では、図２に示すように、順序処理判定ステップ１１，分割判定ステップ１２，信号情報取得ステップ１３，信号代入部分割ステップ１４，変換判定ステップ１５および変換実行ステップ１６が順次実行される。順序処理判定ステップ（順序処理判定部）１１では、図9に示すような回路データベース（ＨＤＬデータベース）の中から順序処理記述を探し、順序処理記述が有るか否かを判定する。

【００３１】分割判定ステップ（分割判定部）１２では、順序処理判定ステップ１１により順序処理記述が有ると判定された場合に、その順序処理記述の中に、信号変化（クロック信号の立ち上がり）に応じて同期処理される部分が存在することを条件にして順序処理記述を分割可能か否か（分割条件を満たしているか否か）を判定し、その部分を同期部として抽出する。抽出されたデータ（ｉｄ情報）は、例えば図４に示すように、メモリ（図示せず）上の同期部抽出データテーブル２１に格納される。

【００３２】信号情報取得ステップ（信号情報取得部）１３では、分割判定ステップ１２で抽出された同期部内において信号代入を行なう部分を信号代入部として抽出し、その信号代入部についての信号情報を取得する。取得されたデータ（ｉｄ情報）は、例えば図５に示すように、メモリ（図示せず）上の信号情報取得データテーブル２２に格納される。

【００３３】信号代入部分割ステップ（分割部）１４では、順序処理記述において、信号情報取得ステップ１３で抽出された信号代入部（信号代入ブロック）とその他の処理部とを、後述するごとく新たな信号を介して分割する。変換判定ステップ（変換判定部）１５では、信号情報取得ステップ１３で取得された信号代入部についての信号情報に基づいて、信号代入部分割ステップ１４で分割された信号代入部を、図１２（ａ）に示すフリップフロップ（ハードウェア素子）３１に置き換え可能か否かを判定する。

【００３４】そして、変換実行ステップ（変換実行部）１６では、変換判定ステップ１５により置き換え可能であると判定された場合に、信号代入部分割ステップ１４で分割された信号代入部を、実際に図１２（ａ）に示すフリップフロップ３１に置き換える。以下に、上述した各ステップ１１～１６のより詳細な処理手順について、それぞれ図３～図８を参照しながら説明する。

【００３５】図3は、順序処理判定ステップ１１の詳細な処理フローを示す図であり、この図3に示すように、順序処理判定ステップ１１では、回路データベースの全ての処理部（ブロック）について順序処理部の探索を行なったか否かを判定し（ステップＳ１１）、未判定の処理部があれば（ステップＳ１１でＮＯ判定の場合）、その処理部が順序処理記述であるか否かを判定する（ステップＳ１２）。順次処理記述でなければ（ＮＯ判定の場合）、次の処理部を探索し（ステップＳ１３）、ステップＳ１１に戻る。

【００３６】そして、回路データベースの全ての処理部について順次処理部の探索を行ない順次処理部が見つけられなかった場合には偽を返し（ステップＳ１１のＹＥＳルートからステップＳ１４）、順序処理判定ステップ１１を終了する。一方、回路データベース中に1つでも

順次処理部が存在する場合には真を返し（ステップＳ１２のＹＥＳルートからステップＳ１５）、順序処理判定ステップ１１を終了する。

【００３７】図４は、分割判定ステップ１２の詳細な処理フローおよび同期部抽出データの具体例を示す図であり、順序処理判定ステップ１１から真が返されてきた場合、この図４に示すように、分割判定ステップ１２では、まず、順序処理記述中に信号変化の判定部（例えば図９に示す機能名“RISE”のブロック）が存在するか否かを判定する（ステップＳ２１）。

【００３８】信号変化の判定部が無ければ偽を返して（ステップＳ２１のＮＯルートからステップＳ２６）、分割判定ステップ１２を終了する。これに対し、信号変化の判定部が有れば（ステップＳ２１でＹＥＳ判定の場合）、その信号変化の判定に用いられるクロック信号を得て、そのクロック信号の識別子を同期部抽出データテーブル２１のデータ“SYNC_PART.SIGNAL”として格納する（ステップＳ２２）。図９に示す例では、機能名“RISE”のブロック（識別子２－１）が信号変化の判定部であり、その信号変化の判定対象のクロック信号は、信号名CLK で識別子１－１の信号である。従って、図４に示すように、同期部抽出データテーブル２１にはデータ“SYNC_PART.SIGNAL”として“１－１”が格納される。

【００３９】クロック信号を得ると、そのクロック判定部から分岐して処理される部分（同期部）を見つけ出し、その同期部の始まりを得て、その始まりのブロックの識別子を同期部抽出データテーブル２１のデータ“SYNC_PART.BODY”として格納する（ステップＳ２３）。図９に示す例では、機能名“RISE”のブロック直後のブロックが同期部の始まりであり、そのブロックの識別子は２－２である。従って、図４に示すように、同期部抽出データテーブル２１にはデータ“SYNC_PART.BODY”として“２－２”が格納される。

【００４０】そして、上述したステップＳ２２およびＳ２３の処理により、“SYNC_PART”の２つのデータが揃った場合には真を返して（ステップＳ２４のＹＥＳルートからステップＳ２５）、分割判定ステップ１２の処理を終了する一方、その２つのデータが何らかの理由によって揃わなかった場合には偽を返して（ステップＳ２４のＮＯルートからステップＳ２６）、分割判定ステップ１２の処理を終了する。

【００４１】図５は、信号情報取得ステップ１３の詳細な処理フローおよび信号情報取得データの具体例を示す図であり、この図５に示すように、信号情報取得ステップ１３では、まず、同期部抽出データテーブル２１におけるデータ“SYNC_PART.BODY”を参照し、その識別子をもつブロックから始まる同期部を探索する（ステップＳ３１）。

【００４２】その同期部の最後まで探索を行なったか否かを判定し（ステップＳ３２）、その同期部についての探索を終了していなければ（ＮＯ判定の場合）、その同期部に信号代入部（例えば図９に示す機能名“ASSIGN”のブロック）が有るか否かを判定する（ステップＳ３３）。信号代入部が無ければ（ＮＯ判定の場合）、ステップＳ３１に戻る一方、信号代入部が有れば（ＹＥＳ判定の場合）、その信号代入部に関するデータ（後述）を取得して信号情報取得データテーブル２２に格納してから（ステップＳ３４）、ステップＳ３１に戻る。このような処理が、同期部の全てについて探索を終了するまで（ステップＳ３２でＹＥＳ判定となるまで）、繰り返し実行され、同期部の全てについて探索を終了すると、信号情報取得ステップ１３の処理を終了する。

【００４３】なお、ステップＳ３４で取得される信号代入部に関するデータとしては、その信号代入部の同期基準となるクロック信号の識別子と、その信号代入部への入力信号を送るブロックの識別子と、その信号代入部からの出力信号を受けるブロックの識別子と、その信号代入部のブロックの識別子とがあり、これらのデータがそれぞれ信号情報取得データテーブル２２のデータ“ASSIGN.SIGNAL”，”ASSIGN.IN_SIGNAL”，”ASSIGN.OUT_SIGNAL”，”ASSIGN.ASSIGN”として格納される。

【００４４】図９に示す例では、信号代入部（“ASSIGN”）の同期基準となるクロック信号は識別子１－１の信号であり、信号代入部への入力信号は識別子２－３のブロック（“OR”）から送られ、信号代入部からの出力信号は識別子１－４のブロック（“Ｐ”）へ送られ、信号代入部のブロックの識別子は２－４である。従って、図５に示すように、信号情報取得データテーブル２２には、データ“ASSIGN.SIGNAL”，”ASSIGN.IN_SIGNAL”，”ASSIGN.OUT_SIGNAL”，”ASSIGN.ASSIGN”として“１－１”，“２－３”，“１－４”，“２－４”がそれぞれ格納される。

【００４５】図６は、信号代入部分割ステップ１４の詳細な処理フローおよび信号情報取得データの具体例を示す図であり、この図６に示すように、信号代入部分割ステップ１４では、まず、信号情報取得データテーブル２２のデータ”ASSIGN.OUT_SIGNAL”を参照し、その信号情報と同じ型の仮の出力信号を作成し、その識別子を信号情報取得データテーブル２２のデータ“ASSIGN.TEMPORARY”として格納する（ステップＳ４１）。図９および図１０に示す例では、仮の出力信号として、信号名“Ｐ′”で識別子１－５のものを作成し、図６に示すように、信号情報取得データテーブル２２には、データ“ASSIGN.TEMPORARY”として“１－５”が格納される。

【００４６】そして、“ASSIGN.TEMPORARY”で示されるブロックの入力に、”ASSIGN.IN_SIGNAL”で示されるブロックを接続する（ステップＳ４２）。図１０に示す例では、識別子１－５のブロック“Ｐ′”の入力に、識別子２－３のブロック“OR”の出力を接続している。ついで、“ASSIGN.TEMPORARY”で示されるブロックの出力

を、"ASSIGN.ASSIGN"で示されるブロックの入力に接続する（ステップS43）。図10に示す例では、識別子1－5のブロック“P′”の出力を、識別子2－4のブロック“ASSIGN”の入力を接続している。

【0047】これらの接続処理の後、"ASSIGN.IN_SIGNAL"で示されるブロックが、"ASSIGN.ASSIGN"で示されるブロックと同じ順序処理内に含まれているか否かを判定する（ステップS44）。同じ順序処理内に含まれていれば（YES判定の場合）には、"ASSIGN.IN_SIGNAL"で示されるブロックとそのブロックに接続されるブロックとを含む順序処理が、“ASSIGN.TEMPORARY”で示されるブロックを介して、"ASSIGN.ASSIGN"で示されるブロックを含む順序処理から分割され、別の順序処理として扱われる（ステップS45）。これにより、クロック信号に同期して動作する信号代入部が順序処理のその他の部分から分割されることになる。

【0048】図9に示す例では、識別子2－3のブロック“OR”が識別子2－4のブロック“ASSIGN”と同じ順序処理1内に含まれているので、図10に示すように、識別子2－3のブロック“OR”とこれに接続される識別子2－2のブロック“NOT”とを含む順序処理部が、新たに導入した識別子1－5のブロック“P′”を介して、識別子2－4のブロック“ASSIGN”を含む順序処理1から分割され、別の順序処理2として扱われている。

【0049】そして、ステップS44にて、"ASSIGN.IN_SIGNAL"で示されるブロックが、"ASSIGN.ASSIGN"で示されるブロックと同じ順序処理内に含まれていないと判定された場合（NO判定の場合）には、全ての信号代入部に対して、上述したステップS41～S45の処理を行なったか否かを判定し（ステップS46）、未処理の信号代入部が有れば（NO判定の場合）には、ステップS41に戻る一方、全ての信号代入部に対する処理を終えている場合（ステップS46でYES判定の場合）には、信号代入部分割ステップ14の処理を終了する。

【0050】図7は、変換判定ステップ15の詳細な処理フローおよびフリップフロップ変換用データの具体例を示す図である。ここでは、変換対象を図12（a）に示すフリップフロップ31に限定しており、変換判定ステップ15では、信号代入部分割ステップ14で分割された信号代入部を含む順序処理部分が、フリップフロップ31に変換可能であるかを判定している。従って、前述したフリップフロップ変換用データの構造や以下に説明する手順なども、それに基づいている。

【0051】この図7に示すように、変換判定ステップ15では、まず、クロック信号などの判定条件を調べ（ステップS51）、信号情報取得データテーブル22を参照して、信号代入部分割ステップ14で分割された信号代入部を含む順序処理部分が、フリップフロップ31に変換できる形式であるか否かを判定する（ステップS52）。ここで、フリップフロップ31に変換できない形式であると判定された場合（NO判定の場合）には、エラー処理を行なう（ステップS57）。

【0052】ステップS52でフリップフロップ31に変化できる形式であると判定された場合（YES判定の場合）には、信号情報取得データテーブル22のデータ“ASSIGN.SIGNAL”，“ASSIGN.TEMPORARY”，"ASSIGN.OUT_SIGNAL"が、フリップフロップ変換用データテーブル23のデータ"FLIPFLOP.SIGNAL"，"FLIPFLOP.IN_SIGNAL"，"FLIPFLOP.OUT_SIGNAL"としてそれぞれ格納される。図10に示す例、即ち図6に示す信号情報取得データテーブル22の具体例に従うと、図7に示すように、フリップフロップ変換用データテーブル23には、データ"FLIPFLOP.SIGNAL"，"FLIPFLOP.IN_SIGNAL"，"FLIPFLOP.OUT_SIGNAL"として“1－1”，“1－5”，“1－4”がそれぞれ格納される。

【0053】ついで、変換対象の順序処理部分に対する非同期信号（リセット信号やセット信号）が有るか否かを判定する（ステップS54）。HDLによる記述で、例えば、“if(reset = '0') then”や“if(reset = '1') then”という記述が含まれているか否かを判定する。非同期信号が有ると判定された場合（YES判定の場合）、信号の値を条件にして非同期に“0”を信号代入している部分をリセットとみなし、フリップフロップ変換用データテーブル23のデータ"FLIPFLOP.RESET"に、その非同期信号の識別子を格納するとともに、信号の値を条件にして非同期に“1”を信号代入している部分をセットとみなし、フリップフロップ変換用データテーブル23のデータ"FLIPFLOP.SET"に、その非同期信号の識別子を格納する（ステップS55）。

【0054】なお、図9～図11では、リセット信号もセット信号も存在しない例が示されている。また、ステップS55の処理後、“0”や“1”以外の値を代入している部分が有るか否かを判定し（ステップS56）、そのような部分があれば（YES判定の場合）には、エラー処理を行なう（ステップS58）。

【0055】そして、ステップS54で非同期信号が無いと判定された場合や、ステップS56で“0”や“1”以外の値を代入している部分が無いと判定された場合には、全ての信号代入部に対して、上述したステップS51～S58の処理を行なったか否かを判定し（ステップS59）、未処理の信号代入部が有れば（NO判定の場合）には、ステップS51に戻る。一方、全ての信号代入部に対する処理を終えている場合（ステップS59でYES判定の場合）には、変換判定ステップ15の処理を終了する。

【0056】図8は、変換実行ステップ16の詳細な処理フローを示す図であり、この図8に示すように、変換実行ステップ16では、まず、変換対象の順序処理部分に、図12（a）に示すフリップフロップ31を組み込む記述を加える（ステップS61）。そして、フリップ

フロップ変換用データテーブル２３を参照し、そのデータである "FLIPFLOP.SIGNAL", "FLIPFLOP.IN_SIGNAL", "FLIPFLOP.OUT_SIGNAL" を識別子としてもつ信号を、フリップフロップ３１のクロック端子 "clock", 入力端子 "input", 出力端子 "output" にそれぞれ接続する（ステップＳ６２）。図１０に示す例では、フリップフロップ３１のクロック端子 "clock", 入力端子 "input", 出力端子 "output" には、識別子１－１のクロック信号 "CLK", 識別子１－５の挿入信号 "P′", 識別子１－４の出力信号 "P" がそれぞれ接続される。

【００５７】ついで、非同期信号が有るか否かを判定し（ステップＳ６３）、非同期信号が有ると判定された場合（ＹＥＳ判定の場合）つまりフリップフロップ変換用データテーブル２３のデータ "FLIPFLOP.RESET", "FLIPFLOP.SET" に識別子が格納されている場合には、これらの識別子をもつ信号を、フリップフロップ３１のリセット端子 "reset", セット端子 "set" にそれぞれ接続する（ステップＳ６４）。前述した通り、図９～図１１では、非同期信号が存在しない例が示されているので、リセット端子やセット端子の接続処理は行なわれない。

【００５８】そして、ステップＳ６３で非同期信号が無いと判定された場合や、ステップＳ６４での接続処理を行なった後には、信号代入部に組み込まれるべき全てのフリップフロップに対して、上述したステップＳ６１～Ｓ６４の処理を行なったか否かを判定し（ステップＳ６５）、未処理のフリップフロップが有れば（ＮＯ判定の場合）には、ステップＳ６１に戻る。

【００５９】一方、全てのフリップフロップに対する処理を終えている場合（ステップＳ６５でＹＥＳ判定の場合）には、フリップフロップに置き換えられた順次処理部を削除し（ステップＳ６６）、変換判定ステップ１５の処理を終了する。なお、図１０に示す例では順次処理１に対応する部分が削除される。このようにして、例えば図９に示す回路データベースに対して、図２～図８にて説明した処理を施すことにより、図１１に示すように、クロック信号の変化に同期する信号代入処理が、フリップフロップ（識別子３－１のブロック）に置き換えられて同時処理に変換され、同時処理１として扱われることになる。

【００６０】このように、本発明の一実施形態としての順序処理記述の変換方法および装置によれば、ＨＤＬによる順序処理記述が、新たに導入した分割用の信号（"ASSIGN.TEMPORARY" に対応する信号；図９～図１１に示す例では識別子１－５のブロック "P′"）を介して、信号代入部（図１０の順序処理１参照）とそれ以外の部分（図１０の順序処理２参照）とに分割して取り扱われ、順次処理記述の長さを短くできるほか、信号変化

に伴う処理は信号代入部だけになり、それ以外の部分を同時処理できるので、ＨＤＬで記述された回路の動作を並列処理によりシミュレートする際、そのシミュレーション処理を大幅に高速化することができる。

【００６１】また、順次処理記述として残る信号代入部を、その信号代入部と同等の動作を行なうフリップフロップ３１に置き換えることで、並列処理によるＨＤＬシミュレーションに際して順次処理を行なう必要がなくなり、その信号代入部を同時処理（図１１の同時処理１参照）できるので、シミュレーション処理をさらに高速化することができる。

【００６２】ここで、ＶＨＤＬで、本実施形態の手法を用いて順序処理記述を信号代入部と組合せ回路とに分割変換した例を、図１５（ａ），（ｂ）に示す。図１５（ａ）は、変換前のＶＨＤＬによる順序処理記述の具体例を示し、図１５（ｂ）は、図１５（ａ）に示す順序処理記述を本実施形態により変換（同時処理化）した結果を示している。

【００６３】図１５（ｂ）に示す記述において、"qq1" や "qq2" が分割用の信号（"ASSIGN.TEMPORARY" に対応する信号）である。このような分割用の信号を挿入するため、信号代入部分が１段増えるので、ＨＤＬシミュレーション実行時には遅延などが発生しないように考慮することが望ましい。また、分割した組合せ回路は、容易に同時処理化することができるため、順次処理はクロックに同期した信号代入のみとなる。これによって、シミュレーション時間を大きく短縮することができる。信号代入部については、前述した通り、フリップフロップに置き換えることで同時処理可能になり、さらにシミュレーション時間を短縮できる。

【００６４】また、表１にて前述した２種類の回路（No.1, No.2）に上述した本実施形態による変換を施した結果を用いて並列処理によるＨＤＬシミュレーションを行なった際の、シミュレーション時間，発生イベント数の結果を表２に示す。また、回路の大きさの目安として、シミュレーションを実行するＣＡＤアクセラレータ中で使用される素子（プリミティブ）の数についても表２に示す。

【００６５】下記表２に示すように、変換前と変換後とでは、２種類の回路とも、プリミティブ数はそれほど大きく変化していないが、シミュレーション時間は約２割、イベント数は４割も減少している。このことから、本実施形態による変換は、順序処理記述のフリップフロップを含むＨＤＬの並列シミュレーションに極めて有効であることが明らかである。

【００６６】

【表２】

変換結果の比較

回路の種類		No.1	No.2
シミュレーション時間（sec）	変換前	9.7	371.4
	変換後	7.9	303.4
イベントの数	変換前	11204800	1085761263
	変換後	6645512	620521476
プリミティブの数	変換前	30970	201620
	変換後	27848	195236

【００６７】なお、上述した実施形態では、信号代入部を、ハードウェア素子である、図１２（ａ）に示すフリップフロップ３１で置き換える場合について説明したが、このフリップフロップ３１と同等の動作をするソフトウェアを呼び出し、その信号処理部をそのソフトウェアで置き換えるように構成してもよい。このフリップフロップ３１をソフトウェアとして実現する際に参照すべき真理値テーブルを図１２（ｂ）に示す。図１２（ｂ）中、“D”は入力端子、“Q”は出力端子を示している。

【００６８】この場合、変換実行ステップ１６では、図８にて説明したステップＳ６１～Ｓ６４の処理に代えて下記のような処理が実行される。つまり、まず、フリップフロップ３１と同等の動作を行なうソフトウェア〔図１２（ｂ）に示す真理値テーブル〕を呼び出す。そして、フリップフロップ変換用データテーブル２３を参照し、そのデータである"FLIPFLOP.SIGNAL", "FLIPFLOP.IN_SIGNAL", "FLIPFLOP.OUT_SIGNAL"を識別子としてもつ信号と、呼び出されたソフトウェアで表現されたフリップフロップのクロック端子(ck)，入力端子（Ｄ），出力端子（Ｑ）との間で各値の受渡しを行なうように接続する。また、非同期信号が有る場合つまりフリップフロップ変換用データテーブル２３のデータ"FLIPFLOP.RESET", "FLIPFLOP.SET"に識別子が格納されている場合には、これらの識別子をもつ信号と、ソフトウェアで表現されたフリップフロップのリセット端子（reset），セット端子（set)との間で各値の受渡しを行なうように接続する。

【００６９】また、上述した実施形態では、図２に示すように、６つのステップ１１～１６の処理を実行して変換を行なっているが、例えば、変換対象の順序処理記述が、図１０の順序処理１として示すように、信号変化に同期する信号代入のみであることが初めから分かっている場合には、図２におけるステップ１２，１４の処理を省略しステップ１１，１３，１５および１６の処理を実行してもよい。

【００７０】さらに、上述した実施形態では、信号代入部の置き換え対象を、図１２（ａ）に示すハードウェア素子または図１２（ｂ）に示す真理値テーブルでソフトウェアとして表される素子である、フリップフロップ３１のみに限定して説明したが、本発明はこれに限定されるものではなく、その他の機能を持つハードウェア素子またはその素子の機能を実現するソフトウェアを用いてもよい。その場合は、置き換えたい記述が、その素子に対応しているかの判定（変換判定ステップ）と必要な情報の取得（テーブル２３に対応する変換用データ取得）とを行なえばよい。このように、置き換え対象の素子の種類を増やすことにより、順序処理記述から分割された信号代入部を、より確実にハードウェア素子またはソフトウェアに置き換えることができ、信号代入部の同期処理化を促進できる。

【００７１】信号代入部の置き換え対象としては、フリップフロップ３１以外に、例えば、図１３（ａ）～（ｃ）に示すようなシフトレジスタ３２や、図１４（ａ）～（ｃ）に示すようなスキャン付きフリップフロップ３３を採用することができる。ここで、図１３（ａ），（ｂ），（ｃ）には、それぞれ、ハードウェア素子としてのシフトレジスタ３２と、そのシフトレジスタ３２をソフトウェアとして実現する際に用いられる真理値テーブルと、そのシフトレジスタ３２に対応するＨＤＬ記述例とが示されている。

【００７２】また、図１４（ａ），（ｂ），（ｃ）には、それぞれ、ハードウェア素子としてのスキャン付きフリップフロップ３３と、そのスキャン付きフリップフロップ３３をソフトウェアとして実現する際に用いられる真理値テーブルと、そのスキャン付きフリップフロップ３３に対応するＨＤＬ記述例とが示されている。上述した実施形態では、図９に示す回路データベースを処理対象の具体例として説明しているが、本発明は、これに限定されるものでなく、他の種々の順序処理記述を同時処理記述に変換するのに用いられる。

【００７３】

【発明の効果】以上詳述したように、本発明の順序処理記述の変換方法および装置によれば、ＨＤＬによる順序処理記述が信号代入部とそれ以外の部分とに分割して取り扱われ、順次処理記述の長さを短くできるほか、信号変化に伴う処理は信号代入部だけになり、それ以外の部分を同時処理化できるので、ＨＤＬで記述された回路の

動作を並列処理によりシミュレートする際に、その処理を大幅に高速化できる効果がある（請求項1，4）。

【0074】また、順次処理記述として残る信号代入部を、その信号代入部と同等の動作を行なうハードウェア素子もしくはソフトウェアに置き換えることで、並列処理によるシミュレーションに際して順次処理を行なう必要がなくなり、さらなる高速化に寄与する（請求項2，3，5，6）。

【図面の簡単な説明】

【図1】本発明の原理ブロック図である。

【図2】本発明の一実施形態としての順序処理記述の変換方法の全体的処理フローおよびその変換装置の構成を示す図である。

【図3】本実施形態における順序処理判定ステップ（順序処理判定部）の詳細な処理フローを示す図である。

【図4】本実施形態における分割判定ステップ（分割判定部）の詳細な処理フローおよび同期部抽出データの具体例を示す図である。

【図5】本実施形態における信号情報取得ステップ（信号情報取得部）の詳細な処理フローおよび信号情報取得データの具体例を示す図である。

【図6】本実施形態における信号代入部分割ステップ（分割部）の詳細な処理フローおよび信号情報取得データの具体例を示す図である。

【図7】本実施形態における変換判定ステップ（変換判定部）の詳細な処理フローおよびフリップフロップ変換用データの具体例を示す図である。

【図8】本実施形態における変換実行ステップ（変換実行部）の詳細な処理フローを示す図である。

【図9】本実施形態の処理対象となる回路データベースの具体例を示す図である。

【図10】本実施形態において信号代入部を分割した結果得られる回路データベースを示す図である。

【図11】本実施形態において信号代入部をフリップフロップにより置き換えた結果得られる回路データベースを示す図である。

【図12】（a），（b）はそれぞれ本実施形態におけるハードウェア素子としてのフリップフロップとそのフリップフロップをソフトウェアとして実現する際の真理値テーブルとを示す図である。

【図13】（a），（b），（c）はそれぞれ本実施形態におけるハードウェア素子としてのシフトレジスタとそのシフトレジスタをソフトウェアとして実現する際に用いられる真理値テーブルとそのシフトレジスタに対応するHDL記述例とを示す図である。

【図14】（a），（b），（c）はそれぞれ本実施形態におけるハードウェア素子としてのスキャン付きフリップフロップとそのスキャン付きフリップフロップをソフトウェアとして実現する際に用いられる真理値テーブルとそのスキャン付きフリップフロップに対応するHDL記述例とを示す図である。

【図15】（a），（b）はそれぞれ変換前の順序処理記述の具体例とその順序処理記述を本実施形態により変換した結果得られた記述とを示す図である。

【図16】VHDLによる順序処理文および同時処理文の具体例を示す図である。

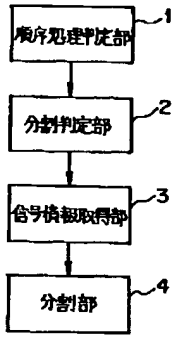
【図17】（a），（b）はそれぞれ従来の手法で同時処理記述に変換できる順序処理文の具体例と従来の手法で同時処理記述に変換できない順序処理文の具体例とを示す図である。

【符号の説明】

1　順序処理判定部
2　分割判定部
3　信号情報取得部
4　分割部
11　順序処理判定ステップ（順序処理判定部）
12　分割判定ステップ（分割判定部）
13　信号情報取得ステップ（信号情報取得部）
14　信号代入部分割ステップ（分割部）
15　変換判定ステップ（変換判定部）
16　変換実行ステップ（変換実行部）
21　同期部抽出データテーブル
22　信号情報取得データテーブル
23　フリップフロップ変換用データテーブル
31　フリップフロップ
32　シフトレジスタ
33　スキャン付きフリップフロップ

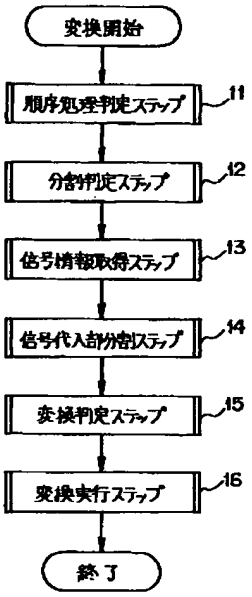
【図1】

本発明の原理ブロック図



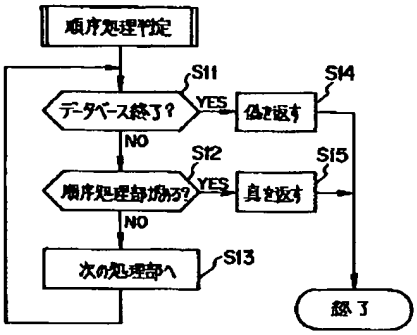
【図2】

本発明の一実施形態としての順序処理記述の変換方法の全体的処理フローおよびその変換装置の構成を示す図



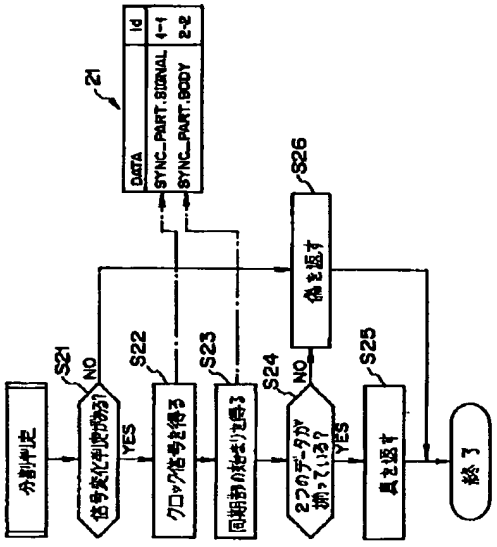
【図3】

本実施形態における順序処理判定ステップ（順序処理判定部）の詳細な処理フローを示す図



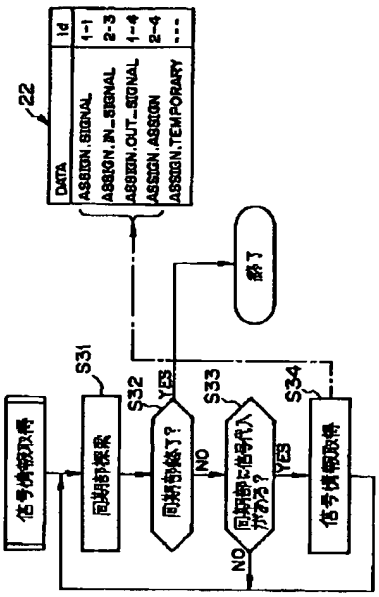
【図4】

本実施形態における分割判定ステップ（分割判定部）の詳細な処理フローおよび同期部抽出データの具体例を示す図



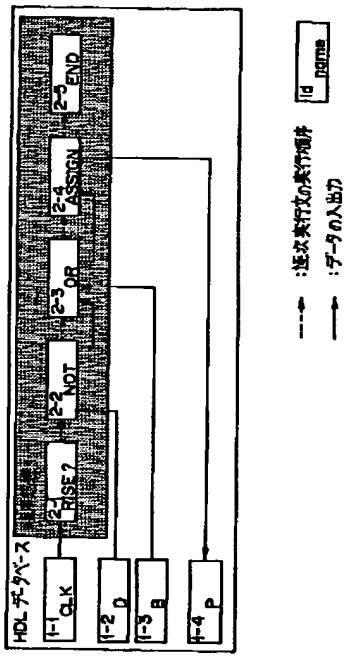
【図5】

本実施形態における信号情報取得ステップ(信号情報取得部)の詳細な処理フローおよび信号情報取得データの具体例を示す図



【図9】

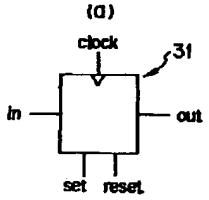
本実施形態の処理対象となる回路データベースの具体例を示す図



【図12】

本実施形態におけるハードウェア素子としてのフリップフロップとそのフリップフロップをソフトウェアとして実現する際の真理値テーブルとを示す図

(a)



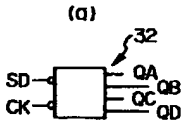
(b)

set	reset	ck	D	Q
L	H	X	X	H
X	L	X	X	L
H	H	L	X	QO
H	H	↑	d	d

【図13】

本実施形態におけるハードウェア素子としてのシフトレジスタとそのシフトレジスタをソフトウェアとして実現する際に用いられる真理値テーブルとそのシフトレジスタに対応するHDL記述例とを示す図

(a)



(b)

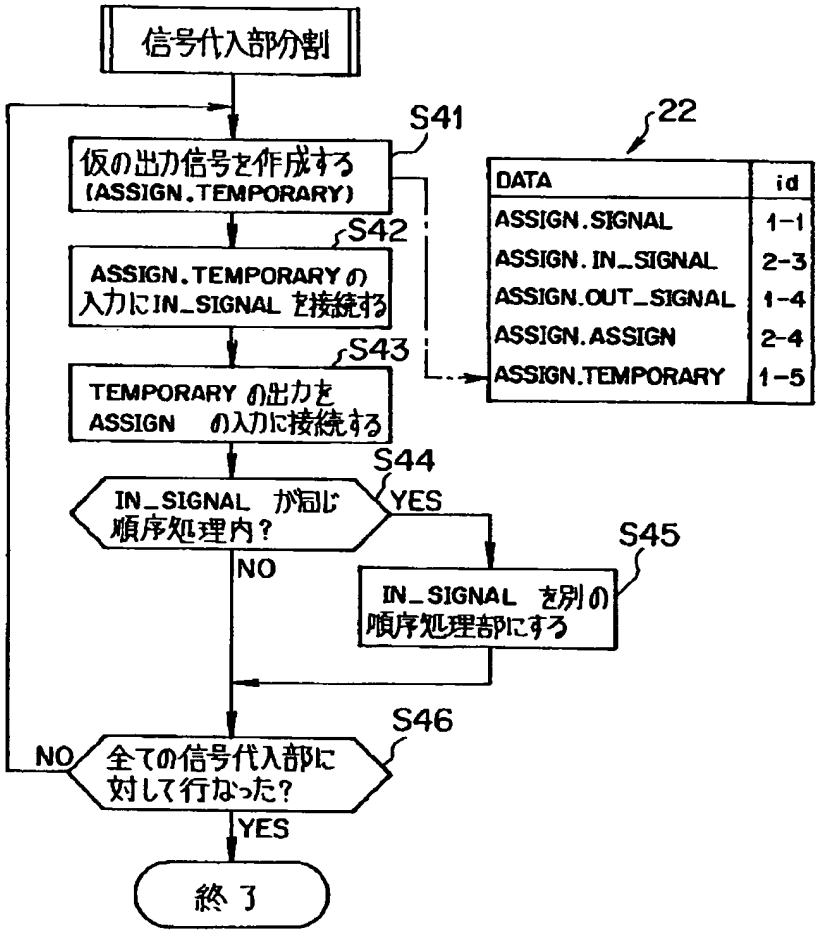
INPUT		OUTPUT			
SD	CK	QA	QB	QC	QD
SD	HtoL	SD	QAn	QBn	QCn

(c)

```
if(CK='0'and CK'event) then
  QD<=QC;
  QC<=QB;
  QB<=QA;
  QA<=SD;
end if;
```

【図６】

本実施形態における信号代入部分割ステップ（分割部）の詳細な処理フローおよび信号情報取得データの具体例を示す図



【図１６】

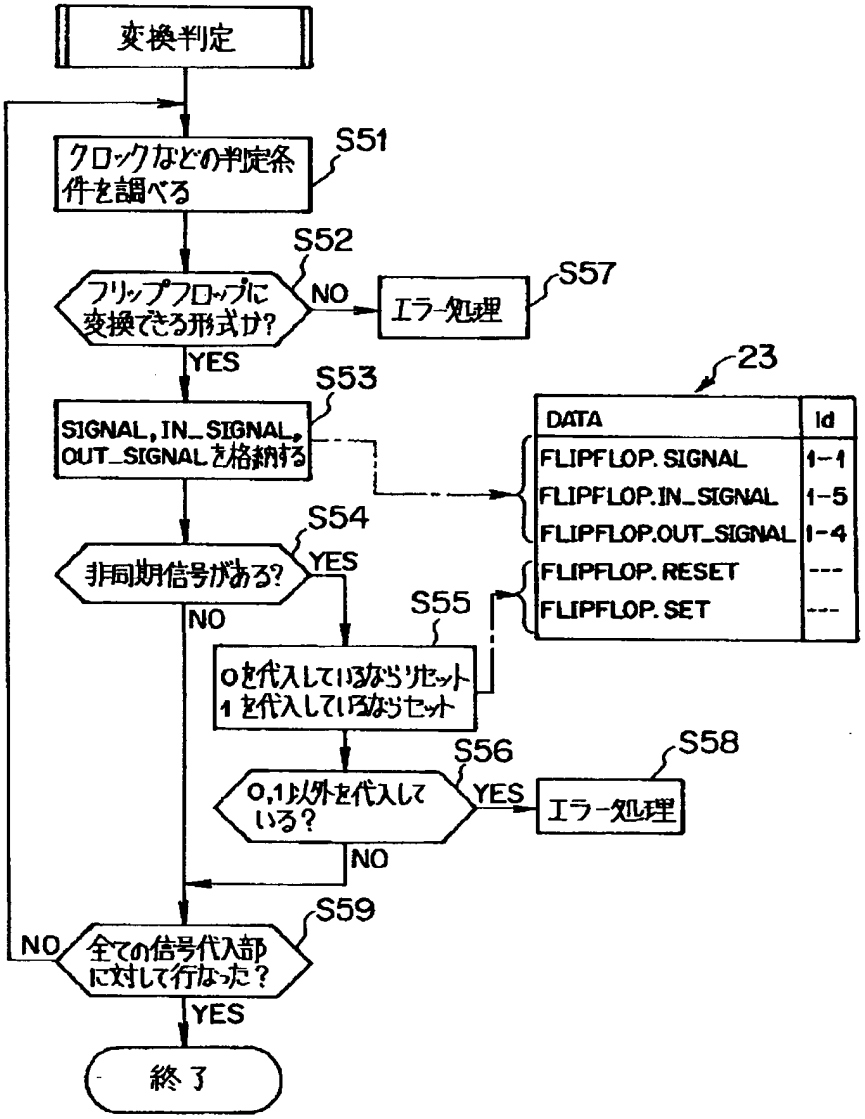
VHDLによる順序処理文および同時処理文の具体例を示す図

```
process (a,b,c,d,e,f)
begin
    sig1 <= a and b;
    sig2 <= c and d;
    sig3 <= e or f;
end process
sig4 <= c when sel2='1' else d;
sig5 <= sig3 and sig4;
```

(1)
(2)
(3)

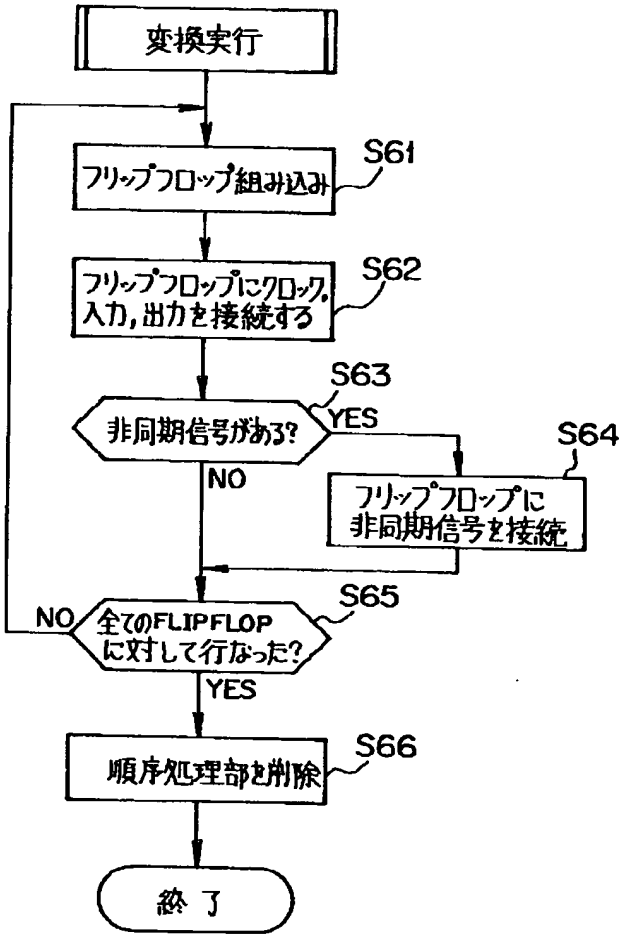
【図7】

本実施形態における変換判定ステップ(変換判定部)の詳細な処理フローおよびフリップフロップ変換用データの具体例を示す図



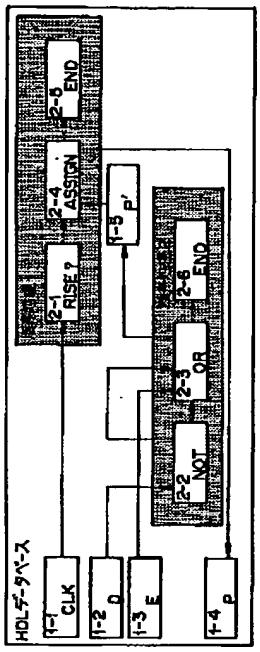
【図8】

本実施形態における変換実行ステップ(変換実行部)の詳細な処理フローを示す図



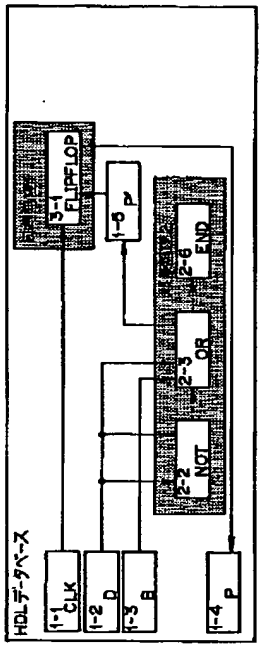
【図10】

本実施形態において信号代入部を分割した結果得られる回路データベースを示す図



【図11】

本実施形態において信号代入部をフリップフロップにより置き換えた結果得られる回路データベースを示す図



【図15】

変換前の順序処理記述の具体例とその順序処理記述を本実施形態により変換した結果得られた記述とを示す図

(a)

```
process (clk)
begin
if (clk='1' and clk'event) then
   q1 <= d1 and e1 and f1;
   q2<= d2 or e2 or f2;
   -----
end if;
end process
```

(b)

```
process(d1,d2,e1,e2,f1,f2)
begin
qq1  = d1 and e1 and f1;
qq2  = d2 or e2 or f2;
  -----
end process
process (clk)
begin
if (clk = '1' and clk'event) then
   q1 <= qq1;
   q2<= qq2;
   -----
end if;
end process
```

組合せ回路部

信号代入部

【図17】

従来の手法で同時処理記述に変換できる順序処理文の具体例と従来の手法で同時処理記述に変換できない順序処理文の具体例とを示す図

(a)

```
if (clk = '1') then
   q1 <= a1 and b1;
   q2<= a2 or b2;
   -----
end if;
```

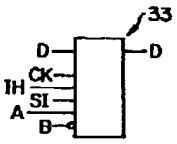
(b)

```
if (clk='1' and clk'event) then
   q1 <= a1 and b1;
   q2<= a2 or b2;
   -----
end if;
```

【図14】

本実施形態におけるハードウェア素子としてのスキャン付きフリップフロップとそのスキャン付きフリップフロップをソフトウェアとして実現する際に用いられる真理値テーブルとそのスキャン付きフリップフロップに対応するHDL 記述例とを示す図

(a)



(b)

MODE	INPUT					OUTPUT
	CLK	D	A	B	SI	Q
CLOCK	LtoH	Di	L	L	X	Di
	H	X	L	L	X	QO
SCAN	H	X	LtoH	H	Si	QO
	H	X	L	HtoL	X	Si

CLK=CK+IH

(c)

```
CLK  CK or IH;
if(CLK='1' and CLK'event and A='0' and B='0') then
    Q<= D;
elsif (A='1' and A'event and CLK='1' and B='1') then
    tmp<= D;
elsif(B='0' and B'event and CLK='1' and A='0') then
    Q<= tmp
end if;
```